Задача A. Конденсация графа

Имя входного файла:	`condense2.in`
Имя выходного файла:	`condense2.out`
Ограничение по времени:	0.65 секунда
Ограничение по памяти:	256 мегабайт

Требуется найти количество рёбер в конденсации ориентированного графа. Примечание: конденсация графа не содержит кратных рёбер и петель.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количество вершин и рёбер графа соответственно ($n \leqslant 10\,000, m \leqslant 100\,000$). Следующие m строк содержат описание рёбер, по одному на строке. Ребро номер i описывается двумя натуральными числами b_i, e_i — началом и концом ребра соответственно ($1 \leqslant b_i, e_i \leqslant n$). В графе могут присутствовать кратные рёбра и петли.

Формат выходных данных

Первая строка выходного файла должна содержать одно число — количество рёбер в конденсации графа.

Примеры

condense2.in	condense2.out
4 4 2 1 3 2 2 3 4 3	2

Задача B. Противопожарная безопасность

Имя входного файла:	`firesafe.in`
Имя выходного файла:	`firesafe.out`
Ограничение по времени:	0.5 секунда
Ограничение по памяти:	256 мегабайт

В Судиславле n домов. Некоторые из них соединены дорогами с односторонним движением.

В последнее время в Судиславле участились случаи пожаров. В связи с этим жители решили построить в посёлке несколько пожарных станций. Но возникла проблема: едущая по вызову пожарная машина, конечно, может игнорировать направление движения текущей дороги, однако возвращающаяся с задания машина обязана следовать правилам дорожного движения (жители Судиславля свято чтут эти правила!).

Очевидно, что, где бы ни оказалась пожарная машина, у неё должна быть возможность вернуться на ту пожарную станцию, с которой она выехала. Но строительство станций стоит больших денег, поэтому на совете посёлка было решено построить минимальное количество станций таким образом, чтобы это условие выполнялось. Кроме того, для экономии было решено строить станции в виде пристроек к уже существующим домам.

Ваша задача — написать программу, рассчитывающую оптимальное положение станций.

Формат входных данных

В первой строке входного файла задано число n ($1 \leqslant n \leqslant 3\,000$). Во второй строке записано количество дорог m ($1 \leqslant m \leqslant 100\,000$). Далее следует описание дорог в формате a_i b_i, означающее, что по i-й дороге разрешается движение автотранспорта от дома a_i к дому b_i ($1 \leqslant a_i, b_i \leqslant n$).

Формат выходных данных

В первой строке выведите минимальное количество пожарных станций K, которое необходимо построить. Во второй строке выведите K чисел в произвольном порядке — дома, к которым необходимо пристроить станции. Если оптимальных решений несколько, выведите любое.

Примеры

firesafe.in	firesafe.out
5 7 1 2 2 3 3 1 2 1 2 3 3 4 2 5	2 4 5

Задача C. Почтальон

Имя входного файла:	`post.in`
Имя выходного файла:	`post.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайта

В городе $\mathfrak{C}$ есть n площадей, соединенных улицами. При этом количество улиц не превышает ста тысяч и существует не более трех площадей, на которые выходит нечетное число улиц. Для каждой улицы известна ее длина. По улицам разрешено движение в обе стороны. В городе есть хотя бы одна улица. От любой площади до любой можно дойти по улицам.

Почтальону требуется пройти хотя бы один раз по каждой улице так, чтобы длина его пути была наименьшей. Он может начать движение на любой площади и закончить также на любой (в том числе и на начальной).

Формат входных данных

Первая строка входного файла содержит натуральное число n — количество площадей в городе ($1 \leqslant n \leqslant 1\,000$). Далее следуют n строк, задающих улицы. В i-й из этих строк находится число m_i — количество улиц, выходящих из площади i. Далее следуют m_i пар положительных чисел. В j-й паре первое число — номер площади, в которую идет j-я улица с i-й площади, а второе число — длина этой улицы.

Между двумя площадями может быть несколько улиц, но не может быть улиц с площади на нее саму.

Все числа во входном файле не превосходят 10^5, общее количество улиц также не превосходит 10^5.

Формат выходных данных

Если решение существует, то в первую строку выходного файла выведите одно число — количество улиц в искомом маршруте (считая первую и последнюю), а во вторую — номера площадей в порядке их посещения.

Если решений нет, выведите в выходной файл одно число `-1`.

Если решений несколько, выведите любое.

Примеры

post.in	post.out
4 2 2 1 2 2 4 1 2 4 4 3 5 1 1 2 2 5 4 8 2 3 8 2 4	5 1 2 4 3 2 1

Задача D. Таня и пароль

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Пока папа был на работе, маленькая девочка Таня решила поиграть с папиным паролем к секретной базе данных. Папин пароль представляет собой строку, состоящую из $n+2$ символов. Она выписала все возможные n трёхбуквенных подстрок пароля на бумажки, по одной на каждую бумажку, а сам пароль выкинула. Каждая трёхбуквенная подстрока была выписана на бумажки столько раз, сколько она встречалась в пароле. Таким образом, в итоге у Тани оказалось n бумажек.

Потом Таня поняла, что папа расстроится, если узнает о ее игре, и решила восстановить пароль или, по крайней мере, хотя бы какую-то строку, соответствующую получившемуся набору трёхбуквенных строк. Вам предстоит помочь ей в этой непростой задаче. Известно, что папин пароль состоял из строчных и заглавных букв латинского алфавита, а также из цифр. Строчные и заглавные буквы латинского алфавита считаются различными.

Формат входных данных

В первой строке следует целое число n ($1 \leqslant n \leqslant 2 \cdot 10^5$), количество трёхбуквенных подстрок, которые получились у Тани.

Следующие n строк каждая содержат по три буквы, образующие подстроку пароля папы. Каждый символ во вводе — строчная или заглавная буква латинского алфавита или цифра.

Формат выходных данных

Если во время игры Таня что-то напутала, и строк, соответствующих данному набору подстрок, не существует, то выведите «`NO`».

Если же возможно восстановить строку, соответствующую данному набору подстрок, то выведите «`YES`», а затем любой подходящий вариант пароля.

Примеры

стандартный ввод	стандартный вывод
5 aca aba aba cab bac	YES abacaba
4 abc bCb cb1 b13	NO
7 aaa aaa aaa aaa aaa aaa aaa	YES aaaaaaaaa

Задача E. 2-SAT

Имя входного файла:	`2sat.in`
Имя выходного файла:	`2sat.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Формулировка `2-SAT`: нужно подобрать значения n булевых переменных так, чтобы все m утверждений вида $x_{i_1} = e_1 \vee x_{i_2} = e_2$ обратились в истину. В данной задаче вам гарантируется, что решение существует.

Формат входных данных

Входной файл состоит из одного или нескольких тестов.

Каждый тест описывается следующим образом. На первой строке число переменных n и число утверждений m. Каждая из следующих m строк содержит числа i_1, e_1, i_2, e_2, задает утверждение $x_{i_1} = e_1 \vee x_{i_2} = e_2$ ($0 \leqslant i_j < n$, $0 \leqslant e_j \leqslant 1$). Ограничения: сумма всех n не больше 100 000, сумма всех m не больше 300 000.

Формат выходных данных

Для каждого теста выведите строку из n нулей и единиц — значения переменных. Если у данной задачи `2-SAT` есть несколько решений, выведите любое.

Примеры

2sat.in	2sat.out
1 0 2 2 0 0 1 0 0 1 1 1 3 4 0 1 1 0 0 0 2 1 1 1 2 0 0 0 0 1	0 01 000

Задача F. Студентам — бесплатно!

Имя входного файла:	`students-free.in`
Имя выходного файла:	`students-free.out`
Ограничение по времени:	0.5 секунд
Ограничение по памяти:	64 мегабайта

Зал Большого галактического театра состоит из S рядов, по S мест в каждом ряду. Продажа билетов на каждый спектакль происходит по следующему принципу: первые $S^2 - N$ ценителей прекрасного приобретают билеты на любые места по их вкусу, а оставшиеся N кресел администрация бесплатно выделяет студентам, отдавая дань сложившимся традициям.

Во избежание обвинений в дискриминации по половому признаку, рассаживать студентов по этим N местам необходимо таким образом, чтобы:

- в каждом ряду количество девушек-студенток и количество юношей-студентов различалось бы не более чем на 1;
- на каждой "вертикали мест" (т. е. местах, имеющих один и тот же номер, но расположенных в разных рядах) количество девушек-студенток и количество юношей-студентов также различалось бы не более чем на 1.

Таким образом, после продажи билетов ценителям прекрасного билетёры должны распределить оставшиеся N кресел на женские и мужские с соблюдением этих правил. Каждое место в зале определяется двумя числами от 1 до S — номером ряда и номером самого места в этом ряду. Студенческое кресло номер i расположено в a_i-м ряду и имеет в нём номер b_i. Поскольку ценители прекрасного могли занять совершенно любые места, числа a_i и b_i могут принимать любые значения от 1 до S. В частности, может оказаться так, что в каком-нибудь ряду не будет ни одного студенческого места.

Ради упрощения работы билетёров администрация обращается к вам с заданием написать программу, которая автоматизирует процесс распределения студенческих мест на мужские и женские.

Формат входных данных

Сначала вводятся два целых числа S и N ($1 \leqslant S \leqslant 100\,000, 1 \leqslant N \leqslant \min\{100\,000, S^2\}$). Далее расположены N пар натуральных чисел (a_i, b_i), не превосходящих S. Гарантируется, что все места различные.

Формат выходных данных

Если искомого способа не существует, выведите `Impossible`. Иначе выведите единственную строку из N символов `M` (мужское) и `W` (женское). Символ на i-й позиции соответствует статусу i-го места в той же нумерации, в которой они были перечислены во входных данных.

Примеры

students-free.in	students-free.out
2 2 2 1 1 2	WM
3 5 1 2 2 3 1 3 2 1 1 1	MMWWM

Задача G. Сигнализация

Имя входного файла:	`alarm.in`
Имя выходного файла:	`alarm.out`
Ограничение по времени:	4 секунды
Ограничение по памяти:	512 мегабайт

Подземный бункер состоит из n комнат, соединённых $n-1$ коридорами. Каждый коридор соединяет две различные комнаты и имеет определённую длину. Бункер устроен таким образом, что из любой комнаты i можно дойти в любую другую комнату j. Заметим, что существует единственный такой путь, не проходящий по одному и тому же коридору дважды. Сумма длин коридоров, составляющих этот путь, называется расстоянием между комнатами i и j и обозначается $\rho(i, j)$.

Каждая комната бункера оборудована звуковой сигнализацией, состоящей из сирены и датчика звука, который её включает. Сирена, включённая в комнате i, активирует датчик звука в каждой комнате, расстояние до которой не превосходит расстояние d_i, определяемое мощностью этой сирены. Другими словами, включение сирены в комнате i автоматически включает сирену во всех комнатах j, таких что $\rho(i, j) \leqslant d_i$. Эта сирена, в свою очередь, может вызвать автоматическое включение других сирен и так далее.

В случае возникновения чрезвычайной ситуации некоторые сирены необходимо включить вручную, после чего звук от них автоматически включит сирены в других комнатах. Правила безопасности предписывают выбор такого набора сирен для ручного включения, который в конце концов приведёт к автоматическому включению сирен во всех комнатах.

Требуется написать программу, которая определяет минимальное количество сирен в наборе, удовлетворяющем правилам безопасности.

Формат входных данных

Первая строка входных данных содержит единственное число n — количество комнат ($1 \leqslant n \leqslant 3000$).

Вторая строка содержит последовательность из n целых чисел d_i, i-е из них равно максимальному расстоянию, на котором расположенная в комнате i сирена активирует датчики ($0 \leqslant d_i \leqslant 10^9$).

Последующие $n-1$ строк описывают коридоры бункера. В i-й из них находятся три целых числа: u_i, v_i, l_i, где u_i, v_i — номера различных комнат, соединённых коридором i, а l_i — длина этого коридора ($1 \leqslant u_i, v_i \leqslant n$; $1 \leqslant l_i \leqslant 10^9$).

Формат выходных данных

Выходные данные должны состоять из единственного числа — минимального количества сирен, которые необходимо включить вручную.

Примеры

alarm.in	alarm.out
10 1 2 2 2 6 3 4 5 4 3 1 2 5 2 3 1 2 4 5 4 5 2 4 6 4 4 7 3 1 8 1 8 9 5 8 10 4	3

Задача H. Раскраска в три цвета

Имя входного файла:	`color.in`
Имя выходного файла:	`color.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Петя нарисовал на бумаге n кружков и соединил некоторые пары кружков линиями. После этого он раскрасил каждый кружок в один из трех цветов — красный, синий или зеленый.

Теперь Петя хочет изменить их раскраску. А именно — он хочет перекрасить каждый кружок в некоторый другой цвет так, чтобы никакие два кружка одного цвета не были соединены линией. При этом он хочет обязательно перекрасить каждый кружок, а перекрашивать кружок в тот же цвет, в который он был раскрашен исходно, не разрешается.

Помогите Пете решить, в какие цвета следует перекрасить кружки, чтобы выполнялось указанное условие.

Формат входных данных

Первая строка содержит два целых числа n и m — количество кружков и количество линий, которые нарисовал Петя, соответственно ($1 \leqslant n \leqslant 1\,000$, $0 \leqslant m \leqslant 20\,000$).

Следующая строка содержит n символов из множества {'R', 'G', 'B'} — i-й из этих символов означает цвет, в который раскрашен i-й кружок ('R' — красный, 'G' — зеленый, 'B' — синий).

Следующие m строк содержат по два целых числа — пары кружков, соединенных отрезками.

Формат выходных данных

Выведите в выходной файл одну строку, состоящую из n символов из множества {'R', 'G', 'B'} — цвета кружков после перекраски. Если решений несколько, выведите любое.

Если решения не существует, выведите в выходной файл слово "Impossible".

Примеры

color.in	color.out
4 5 RRRG 1 3 1 4 3 4 2 4 2 3	GGBR
4 5 RGRR 1 3 1 4 3 4 2 4 2 3	Impossible